

QUALITY OF SERVICE RESOURCE MANAGEMENT APPARATUS AND METHOD FOR MIDDLEWARE SERVICES

FIELD OF THE INVENTION

5 **[0001]** The present invention relates to information systems and, more particularly, to Quality of Service (QoS) management in information systems.

BACKGROUND OF THE INVENTION

10 **[0002]** Quality of Service (QoS) management may be implemented in a networked enterprise system to optimize the use of system resources by a plurality of concurrent network applications. Units of work in enterprise systems generally can be considered as either tasks or messages. In an object-oriented system a primitive task may be, for example, a thread of execution of an object. Messages typically are encapsulations of information of various types. Systems
15 tend to be task-oriented or message oriented. Accordingly, known QoS management methods tend to be either task-oriented or message-oriented.

[0003] Although QoS management concepts and frameworks for task-oriented systems and message-oriented systems may be similar, actual implementation techniques tend to be quite different. In task-oriented systems,
20 QoS management techniques typically focus on prioritizing task executions to meet application QoS requirements. In message-oriented systems, QoS management techniques typically focus on quality of message reception, processing, and delivery. Meanings of quality characteristics such as performance and reliability can be slightly different from those in task-oriented
25 systems. For example, message reliability typically means a combination of delivery guarantees, ordering of messages, and duplicate removals. Task reliability, on the other hand, typically means a probability that a task can be successfully executed without failure.

[0004] QoS management is currently used in task-oriented systems to
30 support computational applications for getting get high priority tasks done in a timely manner in distributed environments. However, QoS management for message-oriented enterprise systems has not been given as much attention. Such systems include enterprise systems in which service-oriented architectures

one QoS parameter, establish with the service requester and monitor a QoS contract that includes the QoS parameters, and manage at least one resource of the enterprise system based on the monitoring.

5 **[0009]** The features, functions, and advantages can be achieved independently in various embodiments of the present inventions or may be combined in yet other embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

10 **[0010]** The present invention will become more fully understood from the detailed description and the accompanying drawings, wherein:

[0011] Figure 1 is a diagram of an information system having a service-oriented architecture (SOA) in accordance with one implementation of the present invention;

15 **[0012]** Figure 2 is a partial listing of a document type definition (DTD) in accordance with one implementation of the present invention;

[0013] Figure 3 is a listing of QoS message in accordance with one implementation of the present invention;

20 **[0014]** Figure 4 is a diagram of a QoS management architecture in accordance with one implementation of the present invention;

[0015] Figure 5 is a listing of a QoS Manager interface in accordance with one implementation of the present invention;

25 **[0016]** Figure 6 is a sequence diagram of a success path for establishing a QoS contract in accordance with one implementation of the present invention; and

[0017] Figure 7 is a listing of an abstract Resource class in accordance with one implementation of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

30 **[0018]** The following description of the preferred systems and methods is merely exemplary in nature and is in no way intended to limit the invention, its application, or uses. Although one or more configurations of the present invention are described with reference to a publisher/subscriber enterprise service-oriented

architecture (SOA), the invention is not so limited. Other and additional configurations are contemplated in connection with other types of information systems, including but not limited to other and additional types of service-oriented architectures, enterprise systems and/or networked systems.

5 **[0019]** Generally, a system having a service-oriented architecture (SOA) may be treated as a composition of a collection of services. Each service may make its functionality available via a clearly defined interface. In a SOA, a service typically is a self-describing and open software component. A SOA includes services, their compositions and interactions.

10 **[0020]** An information system having a publish/subscribe SOA in accordance with one embodiment of the present invention is indicated generally in Figure 1 by reference number 20. Publishers 24 publish messages, for example, on a plurality of topics and subscribers 34 are operated, for example, by interested parties subscribing to messages on particular topics.

15 **[0021]** Publication services 38 and subscription services 42 are decoupled through an information broker (InfoBroker) 46. The publishers 24 and subscribers 34 are service requestors (also referred to herein as “clients” and “applications”) relative to the InfoBroker 46 as a service provider. Publication services 38 and subscription services 42 are included in common services 50
20 provided by the InfoBroker 46 to the publishers 24 and subscribers 34. Common services 50 also include security 54, persistence 56, filtering 58, fusion 60, distribution 62 and discovery 64. Additional services 66 such as subscription registration may also be provided. The information broker 46 also provides to the clients 24 and 34 a QoS management service 70 as a common service 50. The
25 information broker 46 provides access to a QoS manager 74 through which a client may negotiate a QoS contract as further described below. The information broker 46 provides quality of service to the clients 24 and 34 via the QoS management service 70 as further described below.

30 **[0022]** In one embodiment, publishers 24 and subscribers 34 can dynamically discover the InfoBroker 46 and use the foregoing services 50. Such services may have different QoS characteristics and interactions of such services may be dynamic and decoupled. Different publishers 24 and subscribers 34 may have different QoS requirements in terms of performance, reliability, timeliness,

and security. For example, some publishers 24 may have higher priority than others and may require their message to be guaranteed delivered with correct ordering in faster response time. Similarly, some subscribers 34 may be more critical than others and thus require shorter delays in receiving messages.

5 **[0023]** The InfoBroker 46, as a service provider, may provide one or more QoS guarantees to one or more publishers 24 and/or subscribers 34. The InfoBroker 46 may restrict one or more behaviors of publishers 24 and/or subscribers 34, for example, rate(s) of receiving or delivering messages per unit of time and/or message payload size(s). To address QoS requirements of a
10 plurality of concurrent clients such as the publishers 24 and subscribers 34, the QoS management service 70 includes such services as admission control 80, prediction 82, resource management 84, monitoring 86, and adaptation 88.

[0024] Publishers 24 and subscribers 34 may express their QoS requirements to the InfoBroker 46 and negotiate for QoS contracts as further
15 described below. A QoS contract may be transient (for example, on a per session basis) or permanent (for example, for a client having a particular role). The InfoBroker 46 provides resources and mechanisms for satisfying a QoS contract with a client. The InfoBroker 46 monitors system conditions during executions of QoS contracts and provides appropriate adaptation services as further described
20 below.

[0025] QoS management may be implemented as described herein, for example, in Java using JDK1.4. In one configuration, the InfoBroker 46 exports the QoS Manager 74 to make available its QoS management service 70 to clients. A client 24 or 34 sends its QoS requirements as an XML message, as
25 further described below, to the QoS Manager 74. Upon receiving an agreed-upon QoS contract as further described below, the client uses the contract as a context for its interactions (e.g., sending and/or receiving messages) with the InfoBroker 46.

[0026] As further described below, the architecture exemplified in
30 Figure 1 provides component services that cover a plurality of aspects of QoS management for the InfoBroker 46 as QoS service provider. For example, the InfoBroker 46 can analyze application QoS requirements and can make admission control decisions based on policies as well as on current and predicted

future system conditions. The InfoBroker 46 can commit system resources and mechanisms to satisfy QoS requirements, and can monitor and adapt system behaviors at runtime.

[0027] Generally, in the system 20, QoS requirements for the services 5 50 are specified in terms of QoS characteristics. A QoS characteristic describes a concrete aspect or constraint understood by both service requesters and service providers. QoS characteristics can be categorized, for example, into four categories: performance, reliability, timeliness and security. A set of QoS parameters can be associated with each category, for example, as follows. 10 Associated with the performance category are: response time, message throughput, payload size, publishing/subscribing volume rate, and end-to-end delay. Associated with the reliability category are: delivery guarantee, duplication elimination, message ordering, loss probabilities, error rate, retry threshold, message persistency and criticality. Associated with the timeliness category are: 15 time-to-live, deadline, constant bit-rate, frame time, and priority. Associated with the security category are message signing and encryption.

[0028] It should be noted that the foregoing classification of QoS characteristics and parameters is exemplary. Other and/or additional characterizations and/or classifications of aspects of QoS are contemplated in 20 other and/or additional configurations of the present invention. Based on the foregoing classification of QoS characteristics and parameters, the inventors have developed an XML-based language whereby a service requester may express one or more QoS requirements as a QoS message.

[0029] Syntax of the XML-based QoS language may be defined, for 25 example, using an XML Schema or a document type definition (DTD), a portion of which is shown in Figure 2 and indicated generally by reference number 100. The DTD 100 includes and incorporates the above described categories of QoS characteristics, indicated by reference number 104, and the above described QoS parameters, indicated by reference number 110. In addition to specifying 30 requirements based on the four QoS characteristics categories 104, a service requester also provides self-description in a profile element 116 that may include, for example, client name and role, message topic and content type.

[0030] In Figure 3 is shown a portion of an exemplary QoS Message 200 configured using the DTD 100. In the message 200, several QoS parameters 110 are specified (*e.g.*, guaranteed-delivery) and several parameters 110 are omitted (*e.g.*, duplication-elimination). Default values could be used (as specified, 5 for example, in the DTD 100) for omitted parameters.

[0031] Generally, a QoS management architecture according to one or more embodiments of the present invention may include component services, their interactions, and interfaces with external services such as real-time hosts and network condition monitoring, for example, through commercial off the shelf 10 (COTS) monitoring tools. One configuration of such an architecture is indicated generally in Figure 4 by reference number 300. Key component services include a QoS Manager 308, an Establishment Service 312, a Policy Manager 316, a Resource Manager 320, a Prediction Service 324, an Operation Service 328, a Maintenance Service 332, a Monitoring Service 336, an Adaptation Service 340, 15 and a Diagnostic Service 344. In Figure 4 are shown interactions among the foregoing services. Interactions 348 between the Monitoring service and the QoS Diagnostic service follow a registration and notification style, while interactions 352 among other services are based on a request and reply style. As further described below, the Diagnostic Service 344 interfaces with external services 20 such as a real-time host 356 and network 360 via commercial off the shelf (COTS) monitoring tools 364.

[0032] The QoS Manager 308 orchestrates establishment and operation of QoS services for clients. The QoS Manager 308 provides an interface for clients to access QoS management services, as previously 25 discussed with reference to Figure 1. The Establishment Service 312 may establish a QoS contract for a client given a QoS requirement expressed in an XML QoS message as further described below. The Establishment Service 312 uses the Policy Manager 316, Prediction Service 324, and Resource Manager 320, also as further described below. If a contract cannot be established, the 30 Establishment Service 312 reports to the QoS Manager 308 that made the establishment request on behalf of the client.

[0033] The Policy Manager 316 checks QoS policies 368 to ensure that parameters in a client's QoS requirement are permitted for the client's role, and if

permitted, what resources and mechanisms are required to satisfy the requirement. The Resource Manager 320 provides resource lifecycle management including reservation, allocation, and release for system resources. Since resources typically are local to a platform (e.g., CORBA, J2EE, .Net) of the host 356, the Resource Manager 320 defines abstract classes for general QoS management functions. Concrete resource classes are implemented for each host platform as further described below.

[0034] The Prediction Service 324 keeps track of system conditions in terms of several key system parameters (e.g., memory usage, CPU load, network delay). The Prediction Service 324 also predicts future system conditions in a small window of time using various prediction algorithms upon request.

[0035] The Operation Service 328 uses an initialization process 330 to initialize resource configuration for a QoS contract and coordinates services during the execution of a QoS contract. The Maintenance Service 332 may maintain one or more key QoS parameters for a QoS contract and may activate the Adaptation Service 340 upon threshold crossings with respect to such parameters. The Monitoring Service 336 samples and aggregates QoS parameter values. The Monitoring Service 336 registers condition predicates with the Diagnostic Service 344, which returns with notifications when the predicates become true, *e.g.*, due to changes in system conditions.

[0036] The Adaptation Service 340 dynamically changes resources and mechanisms 372 in order to restore key QoS parameters within normal ranges. The Adaptation Service 340 may also take actions upon contract violations by clients. Such actions may include, for example, slowing down message acceptance from a client which publishes far beyond its agreed publishing rate. When, for example, an observed parameter returns below its threshold value, the Maintenance Service 332 may request the Adaptation Service 340 to restore a QoS level according to the QoS contract. An adaptation mechanism 372 can be a plug-in. Thus appropriate adaptation mechanisms 372 can be statically configured and dynamically selected based on policies.

[0037] The Diagnostic Service 344 uses formal reasoning models such as causal networks to aggregate low level system signals into attributes on system conditions. The Diagnostic Service 344 takes real-time inputs from

monitoring tools 364, aggregates data on the fly, and stores the data in a repository 376. The Diagnostic Service 344 may also evaluate any predicates on the attributes upon value changes and trigger notifications to interested parties, for example, the Monitoring Service 336.

5 **[0038]** Clients access QoS management services through the QoS Manager 308 by querying a QoS service provider (e.g., an information broker as described with reference to Figure 1) or by using a discovery service. In the present embodiment, the QoS Manager 308 is the only component in the QoS management architecture 300 with which a client directly interfaces for QoS
10 services. An exemplary QoS Manager interface, defined, for example, in Java, is referred to generally in Figure 5 by reference number 400. The interface 400 provides, for example, an operation 408 for establishing a QoS contract, an operation 412 for revising a QoS contract, an operation 416 for agreement on a QoS contract offer, and an operation 420 for aborting establishment of a QoS
15 contract.

[0039] A UML sequence diagram of an exemplary success path for establishing a QoS contract is indicated generally in Figure 6 by reference number 500. Upon receiving a request from a client 502, at step 504 the QoS Manager 308 extracts a QoS message, which includes the client's role,
20 credentials, and QoS requirements as previously discussed with reference to Figures 2 and 3. At step 508 the QoS Manager 308 requests the Establishment Service 312 to try to create a QoS contract that satisfies the requirements appropriate for the client's role.

[0040] At step 512 the Establishment Service 312 interprets the QoS
25 message and requests the Policy Manager 316 to start an admission control process to determine whether the system 300 can admit the client 502 based on its role and requested QoS parameters. If the Policy Manager 316 denies the client's request, the Establishment Service 312 returns a message to the QoS Manager 308, which notifies the client as to the denial. The client 502 has a
30 chance to revise its QoS requirements, for example, via the reviseQosContract operation 412 in the interface 400. If the Policy Manager 316 determines at step 516 that the client's request is consistent with policies of the system, the Policy Manager 316 returns at step 520 a set of resources and mechanisms that are to

be allocated and activated for the client's request to the Establishment Service 312.

5 **[0041]** At step 524 the Establishment Service 312 queries the Prediction Service 324 for a then-current system condition (*e.g.*, lightly loaded, normal, overloaded, *etc.*) and predicted availability of resources in the near future. If at step 528 the Prediction Service 324 indicates that resources are available, at step 532 the Establishment Service 312 requests the Resource Manager 320 to reserve the resources at step 536. If the Resource Manager 320 at step 540 indicates that resources are successfully reserved, the Establishment
10 Service 312 at step 544 returns to the QoS Manager 308 with a QoS contract, which includes resources and mechanisms that meet QoS requirements of the client 502. The QoS Manager 308 forwards the contract to the client 502 at step 548. If the contract does not succeed, the Establishment Service 312 returns exceptions to the QoS Manager 308, which notifies the client 502.

15 **[0042]** At the client's agreement on a QoS contract at step 552 (for example, via the agreeQosContractOffer operation 416 in the interface 400), at step 556 the QoS Manager 308 requests the Operation Service 328 to commit and initialize the appropriate resources and mechanisms, set up monitors for QoS parameters if appropriate, and to connect the Monitoring Service 336 and
20 Adaptation Service 340 with the Maintenance Service 332. At step 560 the Operation Service 328 uses the initialization process 330 to request self-configuration of the appropriate resources, set properties of mechanisms, and activate the resources and mechanisms to start operations. QoS parameter values requested by the client and included in the QoS contract are translated
25 into attribute value settings on allocated resources. It should be noted that the inventors treat mechanisms as active resources, since execution of a mechanism typically requires one or more system resources, for example, CPU time. At step 564 the Operation Service 328 indicates to the QoS manager 308 that resources are committed and initialized. At step 568 the QoS manager 308 forwards the
30 foregoing information to the client 502.

[0043] Generally, to directly support enterprise information services such as publish/subscribe, the QoS Resource Manager 320 directly manages resources at a middleware layer, *e.g.*, message queues and/or delivery threads

used in a publish/subscribe enterprise information architecture. Such resources tend to be middleware-platform-dependent and implementation-specific, e.g., different publish/subscribe services may use different logic to implement message queues and/or message delivery mechanisms. In one configuration of the present invention, the QoS Resource Manager 320 manages both passive resources, e.g., buffers, and active resources, e.g., threads. Active resources may be used to encapsulate mechanisms. Such resources may include algorithms for guaranteed message delivery, removing duplicates, and adaptation. The Resource Manager 320 manages active resources and passive resources in the same or a similar manner, except that active resources may be activated and/or frozen from time to time.

[0044] So that the Resource Manager 320 may be decoupled from a host platform and specific enterprise information services, resources are self-configurable, i.e., a resource “knows” what it is designed to do. For example, a receiver thread resource knows how to configure itself with a message queue so that it can store received messages to the queue. Additionally, a resource complies with a common management interface as further described below, so that the Resource Manager 320 can manage resources without referring to implementation logic for such resources.

[0045] In one configuration, a common management interface is provided using an abstract Resource class indicated generally in Figure 7 by reference number 600. The Resource Manager 320 uses the interface 600, for example, at 604 to create, at 608 to configure, at 612 to commit, at 616 to cancel, at 620 to activate, at 624 to freeze, and/or at 628 to release resources, as well as at 632 to predict OS level resource consumption for each resource. The interface 600 also allows resources at 636 to register monitoring probes and at 640 to expose adaptation mechanisms to the QoS management service 300. Resources provide the implementation of the management interface 600 by subclassing the class represented in the interface 600. Such subclassing can be performed by implementing a resource as an object of a subclass, or by providing a proxy of the resource as an object of a subclass. Additionally, active resources implement an executable interface such as the Java Runnable interface in a J2EE-based middleware platform.

[0046] To support critical service requests, resources may be dedicated to a single QoS contract (for example, for a client with high QoS requirements). To support scalability of enterprise information services, resources may be shared by many different QoS contracts (possibly by different clients). An “attach”
5 method 644 may be used to share an existing resource with a new QoS contract.

[0047] The Resource Manager 320 may use a policy-driven or optimization-based approach to manage the resources, e.g., to make a decision as to whether to create a new resource or share an existing resource, whether to cancel particular resources or commit such resources for use. In a policy-driven
10 approach, the Resource Manager 320 can use a local policy to decide for each requested resource whether to create a new instance or to share an existing one when the Establishment Service 312 asks it to reserve resources for a given QoS contract. If a new instance is preferred or required, the Resource Manager 320 uses the “create” method 604 in the abstract Resource class 600 to create a new
15 resource with configuration information supplied by the particular contract (and possibly modified by the policy). If sharing is preferred, the Resource Manager 320 chooses an existing resource and possibly reconfigures it using new configuration information.

[0048] If contract negotiation fails, the Resource Manager 320 cancels
20 all reservations of appropriate resources. Otherwise, the Resource Manager 320 requests the appropriate resources to commit to a given contract. When a contract is released, the Resource Manager 320 requests the appropriate resources to release themselves. As previously mentioned above, other methods in the interface 600 may not be used directly by the Resource Manager 320 but
25 may be used by other components to operate QoS contracts.

[0049] In one configuration, a plurality of Resource subclasses are implemented. For example, a subclass InCommChannel encapsulates an incoming network communication channel (e.g., socket port). A subclass OutCommChannel encapsulates an out-going network communication channel
30 (e.g., destination IP and port). A subclass MessageQueue stores messages received from InCommChannel. A subclass NotificationQueue stores messages to be delivered with destination information attached to each message. A subclass Receiver thread receives bytes from the InCommChannel, assembles

them into messages and pushes messages into the MessageQueue. A subclass Distributor thread distributes messages to NotificationQueues and assigns appropriate destination information according to message topic and all registered subscriptions. A subclass Deliver thread delivers messages in NotificationQueues
5 to the destination. Receiver, Distributor, and Deliver threads are active resources. Once committed, active resources drive the execution of, for example, a publish/subscribe service for all clients.

[0050] The foregoing various configurations of the present invention may be seen additionally to embody a machine-readable medium for use with a
10 processor having a memory. The machine-readable medium includes instructions to cause a processor to receive a quality of service (QoS) message from a client of an information system expressing one or more QoS requirements as one or more parameter values. The machine-readable medium also includes instructions to cause a processor to establish a contract with the client for quality of service
15 based on the one or more parameter values. Additionally, the machine-readable medium includes instructions to cause a processor to allocate one or more resources of the information system to the client based on the contract.

[0051] In a SOA in which a configuration of the foregoing QoS management system is implemented, an InfoBroker can offer differentiated
20 services to clients of different roles and QoS requirements. The InfoBroker thus can optimize the allocation and utilization of resources to satisfy requirements from many concurrent clients. Highly critical clients get more resources or more shares of resource utilization than less critical clients.

[0052] Configurations of the foregoing system can be used to manage
25 network throughput to provide reliable information delivery based on priorities. Because resources are modeled in an object-oriented approach, the foregoing method enables uniform resource management and polymorphic resource allocation across multiple middleware platforms. The foregoing method for resource allocation is easier to use and is more flexible than other methods
30 currently available. System costs can be reduced while resource needs of critical clients can be satisfied.

[0053] The above-described XML-based QoS language provides a flexible and extensible way for applications to express QoS requirements in

various aspects of QoS characteristics. QoS service providers thus can be provided with an architecture that integrates and processes of all aspects of QoS characteristics in a single framework. In the foregoing system, resources of an information system are modeled as software objects to provide polymorphic
5 resource allocation across multiple middleware platforms. QoS characteristics for both tasks and messages are managed in a single framework. QoS requirements of concurrent applications are satisfied, for example, through a QoS service provider in a system middleware layer.

[0054] Resources are classified into categories based on their utilities,
10 greatly simplifying their configurations at runtime. The foregoing resource allocation method may be policy-based and is easily modified and extensible. The method demonstrably shortens delays for highly critical clients than less critical clients by effectively managing allocated resources. Configurations of the present invention can adapt resource allocations when runtime behaviors of clients
15 change to enforce client QoS contracts.

[0055] While various preferred embodiments have been described, those skilled in the art will recognize modifications or variations which might be made without departing from the inventive concept. The examples illustrate the invention and are not intended to limit it. Therefore, the description and claims
20 should be interpreted liberally with only such limitation as is necessary in view of the pertinent prior art.